From COMpfun Authors: HTTP Statuses and Compromised TLS

Denis Legezo

Senior Security Researcher, Kaspersky

HITB LOCKDOWN 002 livestream

The plan

- Part I: On the fly infection and TLS traffic reading

- Part II: Visa application and rare HTTP statuses

- Part III: Two approaches

- Part IV: Hope we would have time to discuss it all

How it all started

	Initial infection	Escalation, detection	Reductor RAT
Malware	COMpfun trojan	Reductor dropper-decryptor	Reductor trojan
Process	One of the browsers	Same browser	lsass.exe
Persistence	COM CLSID hijacking	Auxiliary module, N/A	LSA notification package
Net encryption	AES 128	Local module, N/A	AES 128
Host encryption	Configuration data under constant one byte XOR + LZNT1	Reductor in resources under constant one byte XOR + LZNT1	Victims' unique IDs in TLS "client hello" under XOR with changing key

Why another trojan?

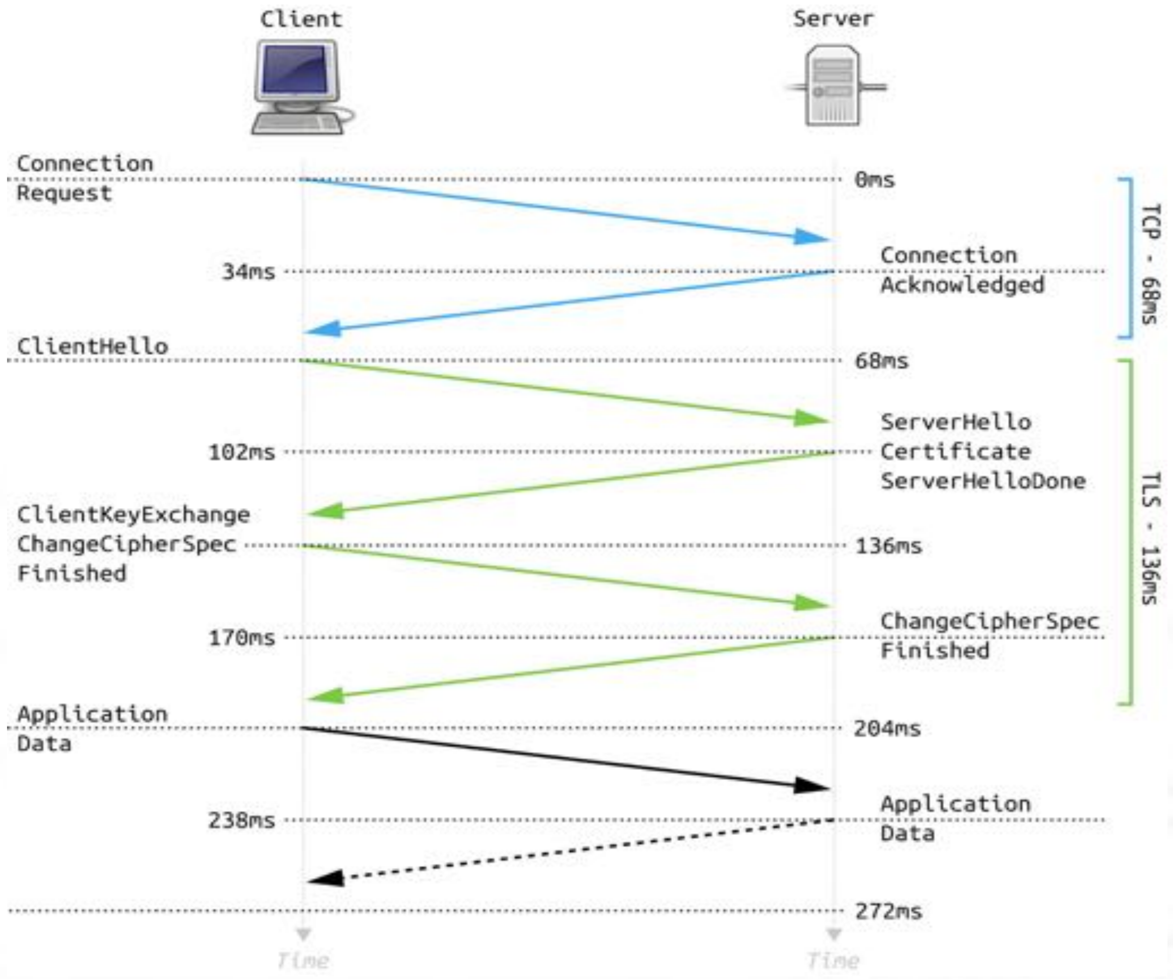
- Keylogging? May be too loud

- Decrypting? May be not in reasonable time with current TLS
- Certificates pre-installation? Could facilitate MITM, but what about NAT?

- Plus marker for packets of interest? Could be next step forward, but too loud again

- Mark TLS session without even single touch of network packets

“Client hello” field



PRNG to mark it

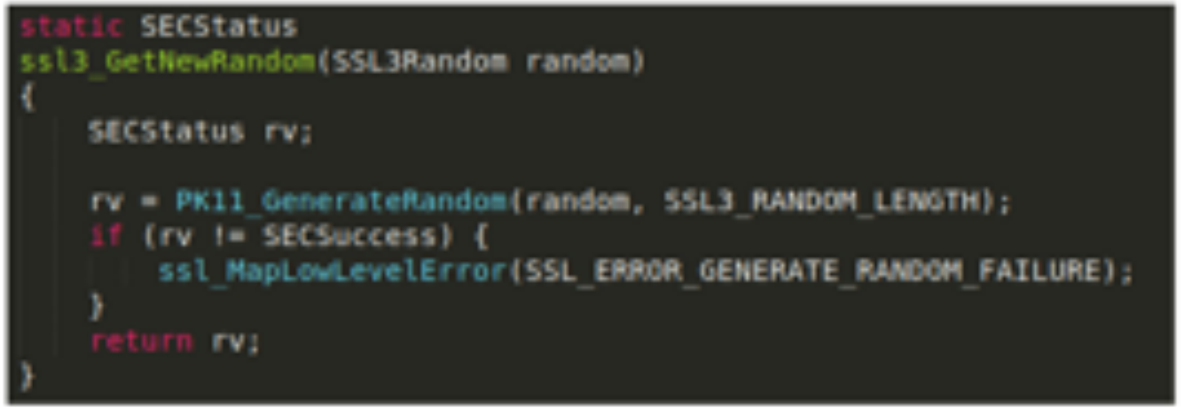
nss3.dll	PK11_GenerateRandom()	Call original PRNG function and generate initial XOR key from its result. Change PRNG result: set seventh byte to 1, then save 0x45F2837D, hwid and cert hashes. Encrypt the result and return it instead of the original PRN. It would affect calls to ssl3_SendClientHello() -> ssl3_GetNewRandom(ss->ssl3.hs.client_random);
advapi32.dll	CryptGenRandom()	Spoof these system PRNG functions result in quite similar way with some minor changes;
bcrypt.dll	BCryptGenRandom()	
chrome.dll	PRNG function	Find PRNG function by its binary code template and patch it like all aforementioned;

Chrome and Firefox

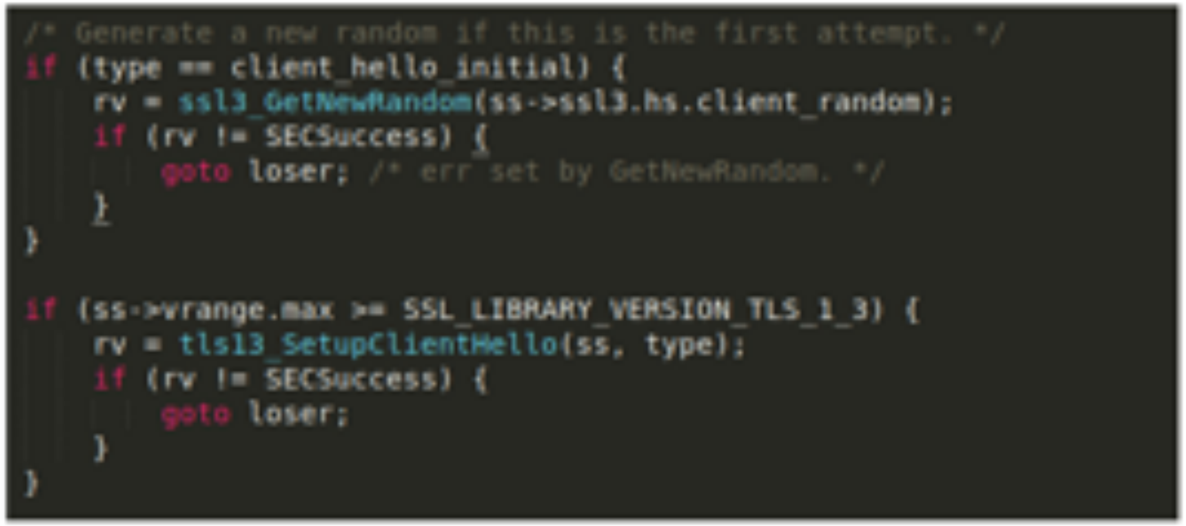
To patch browsers’ PRNG functions in memory and add unique user IDs into TLS handshake developers have to analyze

Firefox sources

Chrome binaries

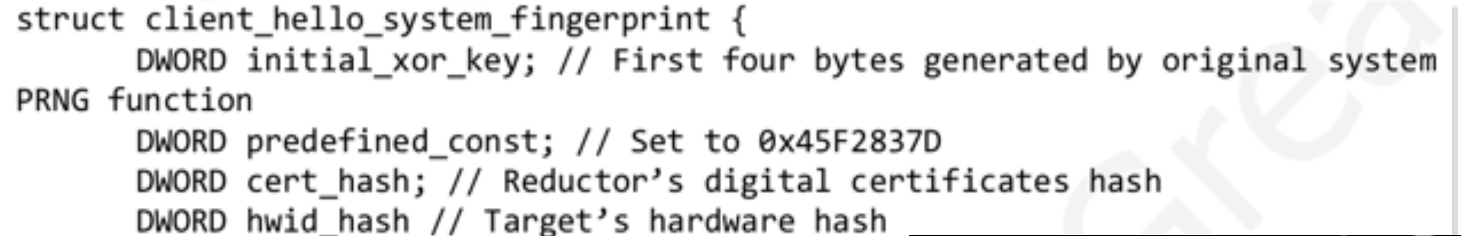


```
static SECStatus
ssl3_GetNewRandom(SSL3Random random)
{
    SECStatus rv;

    rv = PK11_GenerateRandom(random, SSL3_RANDOM_LENGTH);
    if (rv != SECSuccess) {
        ssl_MapLowLevelError(SSL_ERROR_GENERATE_RANDOM_FAILURE);
    }
    return rv;
}
```



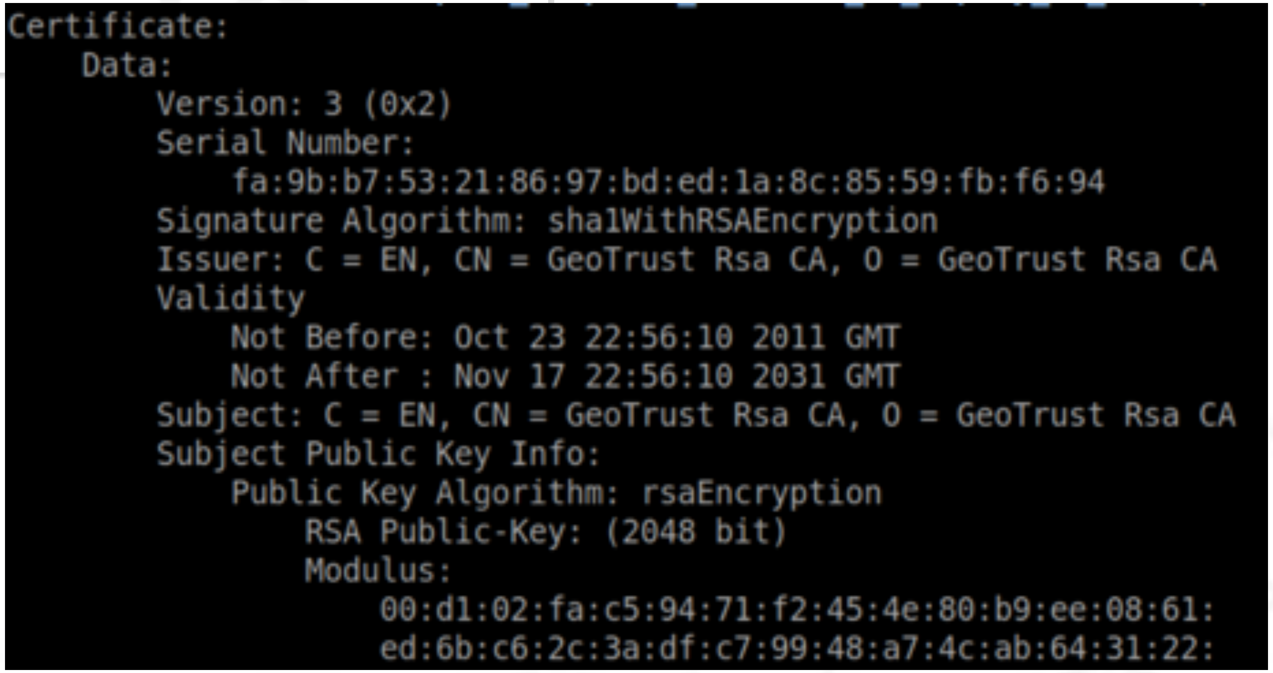
```
/* Generate a new random if this is the first attempt. */
if (type == client_hello_initial) {
    rv = ssl3_GetNewRandom(ss->ssl3.hs.client_random);
    if (rv != SECSuccess) {
        goto loser; /* err set by GetNewRandom. */
    }
}

if (ss->vrange.max >= SSL_LIBRARY_VERSION_TLS_1_3) {
    rv = tls13_SetupClientHello(ss, type);
    if (rv != SECSuccess) {
        goto loser;
    }
}
```

Silently marked



```
struct client_hello_system_fingerprint {
	DWORD initial_xor_key; // First four bytes generated by original system
PRNG function
	DWORD predefined_const; // Set to 0x45F2837D
	DWORD cert_hash; // Reductor's digital certificates hash
	DWORD hwid_hash // Target's hardware hash
};
```

Easter eggs are “UAC is useless” and compfun[.]net domain

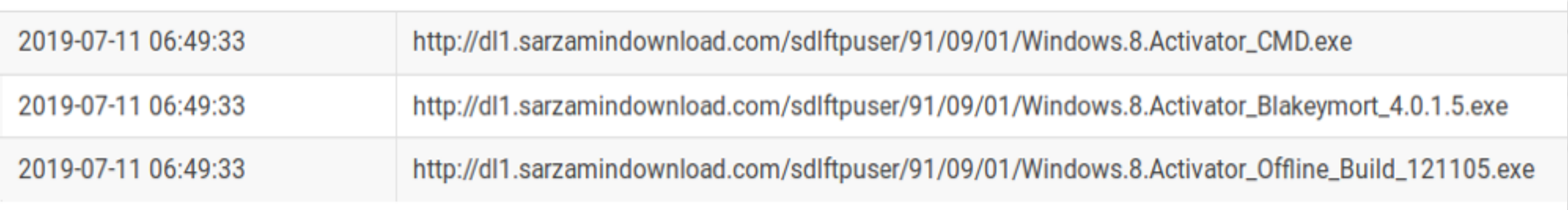


```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            fa:9b:b7:53:21:86:97:bd:ed:1a:8c:85:59:fb:f6:94
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C = EN, CN = GeoTrust Rsa CA, O = GeoTrust Rsa CA
        Validity
            Not Before: Oct 23 22:56:10 2011 GMT
            Not After : Nov 17 22:56:10 2031 GMT
        Subject: C = EN, CN = GeoTrust Rsa CA, O = GeoTrust Rsa CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d1:02:fa:c5:94:71:f2:45:4e:80:b9:ee:08:61:
                    ed:6b:c6:2c:3a:df:c7:99:48:a7:4c:ab:64:31:22:
```

Why on the fly?

Once our telemetry shows new URLs and that time installers were available on the warez web-site

Available and uninfected

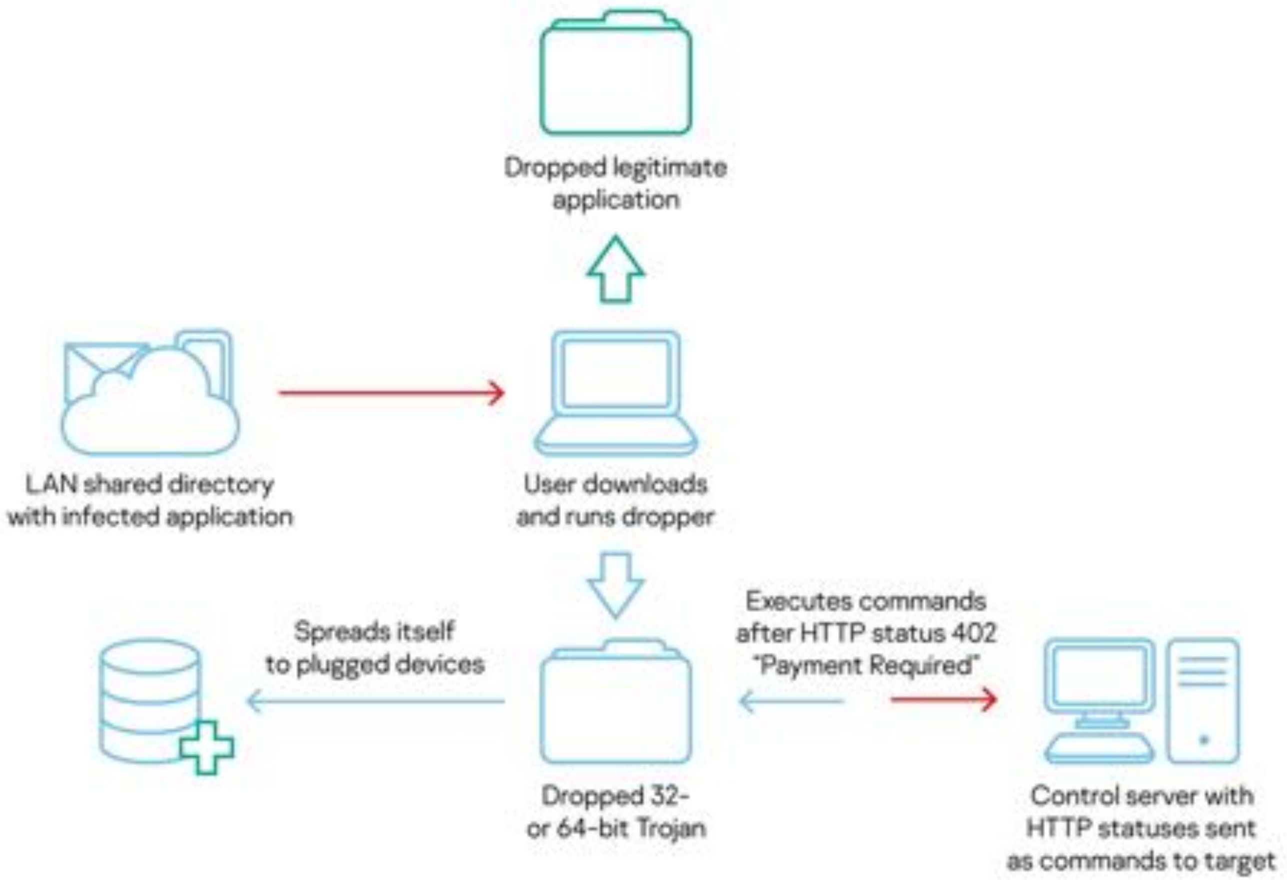


2019-07-11 06:49:33	http://dl1.sarzamindownload.com/sdlftpuser/91/09/01/Windows.8.Activator_CMD.exe
2019-07-11 06:49:33	http://dl1.sarzamindownload.com/sdlftpuser/91/09/01/Windows.8.Activator_Blakeymort_4.0.1.5.exe
2019-07-11 06:49:33	http://dl1.sarzamindownload.com/sdlftpuser/91/09/01/Windows.8.Activator_Offline_Build_121105.exe

Infection chain

Infection chain includes SMB share with spoofed visa application

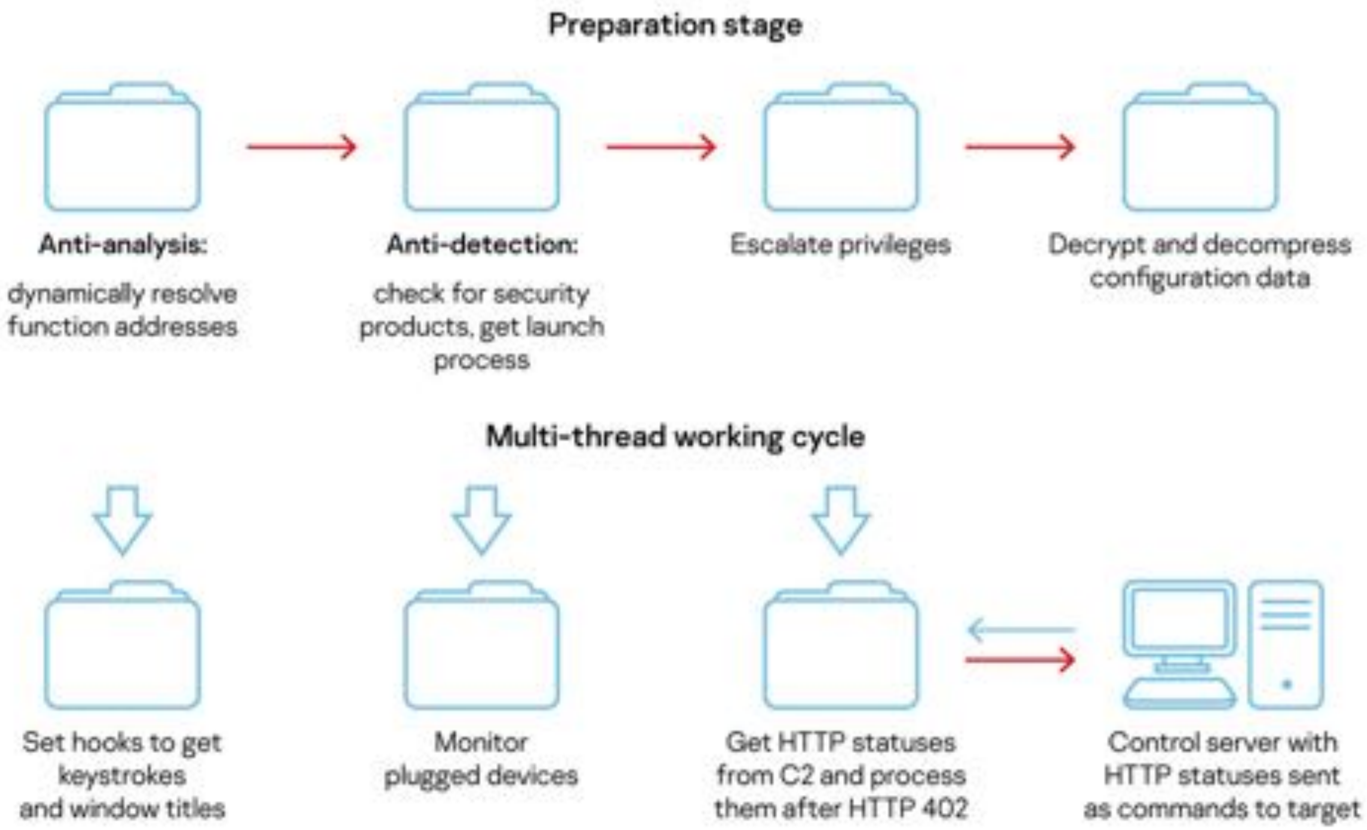
What interested us the most is the C2 command system


Dropped legitimate application
LAN shared directory with infected application
User downloads and runs dropper
Spreads itself to plugged devices
Executes commands after HTTP status 402 "Payment Required"
Dropped 32- or 64-bit Trojan
Control server with HTTP statuses sent as commands to target

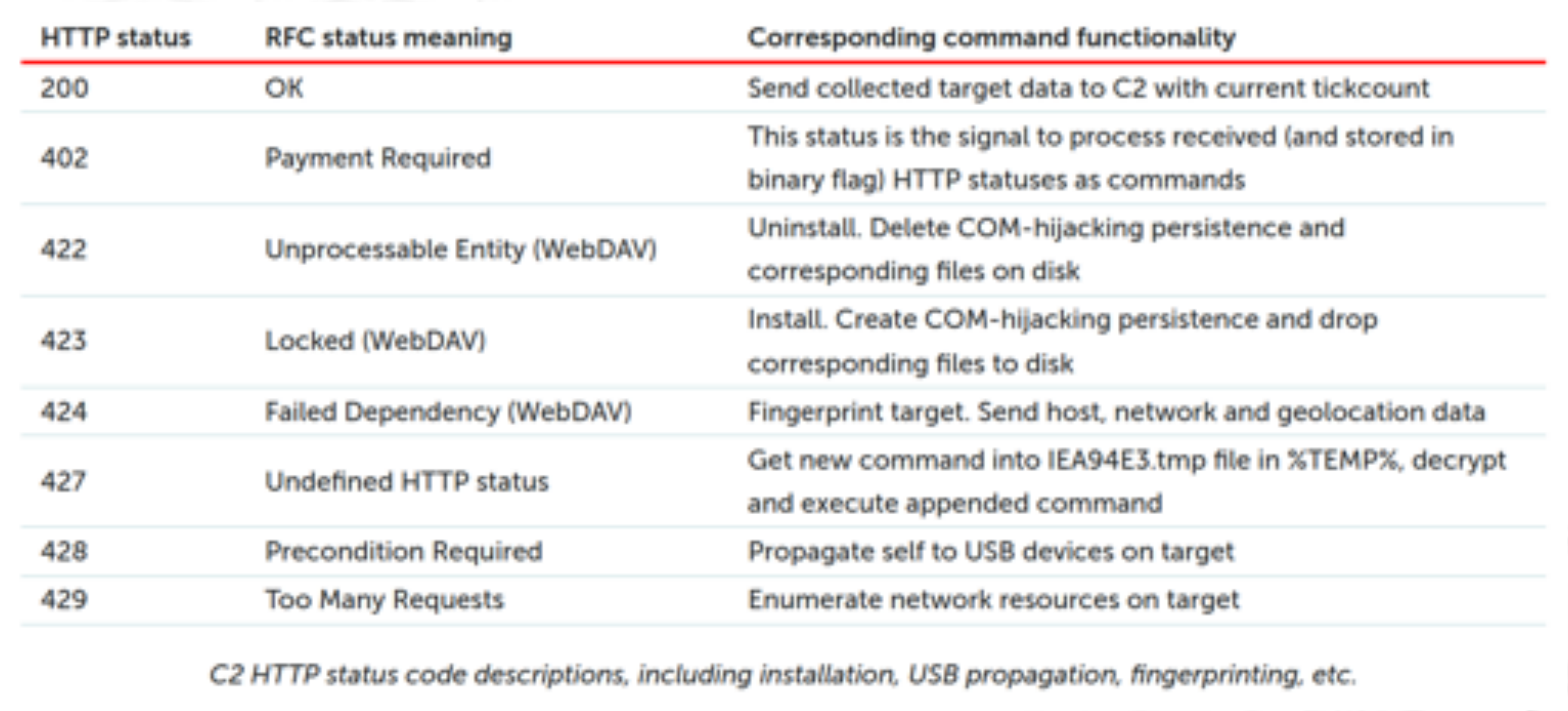
C2 communications

HTTP statuses 422-429 (IETF RFC 7231, 6585, 4918) are the async commands from C2

HTTP status 402 “Payment required” runs handlers


Preparation stage
Anti-analysis:
dynamically resolve function addresses
Anti-detection:
check for security products, get launch process
Escalate privileges
Decrypt and decompress configuration data
Multi-thread working cycle
Set hooks to get keystrokes and window titles
Monitor plugged devices
Get HTTP statuses from C2 and process them after HTTP 402
Control server with HTTP statuses sent as commands to target

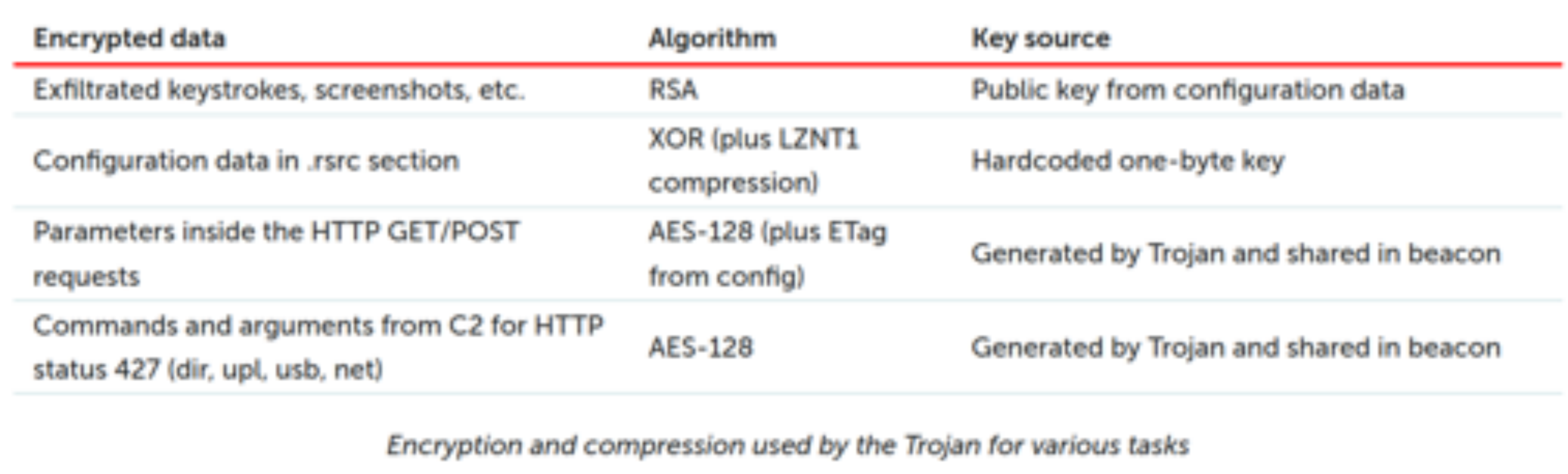
HTTP statuses



HTTP status	RFC status meaning	Corresponding command functionality
200	OK	Send collected target data to C2 with current tickcount
402	Payment Required	This status is the signal to process received (and stored in binary flag) HTTP statuses as commands
422	Unprocessable Entity (WebDAV)	Uninstall. Delete COM-hijacking persistence and corresponding files on disk
423	Locked (WebDAV)	Install. Create COM-hijacking persistence and drop corresponding files to disk
424	Failed Dependency (WebDAV)	Fingerprint target. Send host, network and geolocation data
427	Undefined HTTP status	Get new command into IEA94E3.tmp file in %TEMP%, decrypt and execute appended command
428	Precondition Required	Propagate self to USB devices on target
429	Too Many Requests	Enumerate network resources on target

C2 HTTP status code descriptions, including installation, USB propagation, fingerprinting, etc.

Encryption



Encrypted data	Algorithm	Key source
Exfiltrated keystrokes, screenshots, etc.	RSA	Public key from configuration data
Configuration data in .rsrc section	XOR (plus LZNT1 compression)	Hardcoded one-byte key
Parameters inside the HTTP GET/POST requests	AES-128 (plus ETag from config)	Generated by Trojan and shared in beacon
Commands and arguments from C2 for HTTP status 427 (dir, upl, usb, net)	AES-128	Generated by Trojan and shared in beacon

Encryption and compression used by the Trojan for various tasks

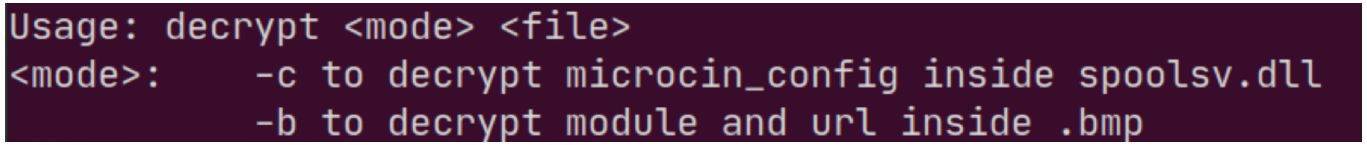
Some math inside

$$\text{timeout} = \sum_{n=0}^{19} \frac{a^n}{n!}$$

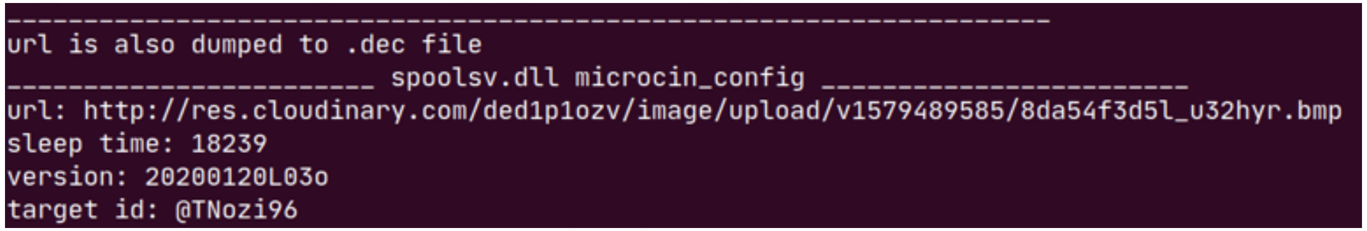
To do or to use?

It you decide to do

In config: version, target ID, URL. Almost certainly constructed with builder



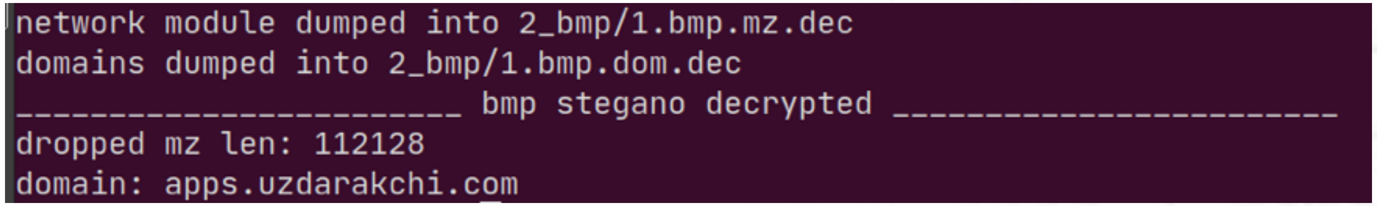
```
Usage: decrypt <mode> <file>
<mode>:     -c to decrypt microcin_config inside spoolsv.dll
            -b to decrypt module and url inside .bmp
```



```
---------------------------------------------------------------------
url is also dumped to .dec file
________________________ spoolsv.dll microcin_config ________________________
url: http://res.cloudinary.com/ded1p1ozv/image/upload/v1579489585/8da54f3d5l_u32hyr.bmp
sleep time: 18239
version: 20200120L03o
target id: @TNozi96
```

In bitmaps: C2 domain and last-stager network module

Key scheduling differs



```
network module dumped into 2_bmp/1.bmp.mz.dec
domains dumped into 2_bmp/1.bmp.dom.dec
________________________ bmp stegano decrypted ________________________
dropped mz len: 112128
domain: apps.uzdarakchi.com
```

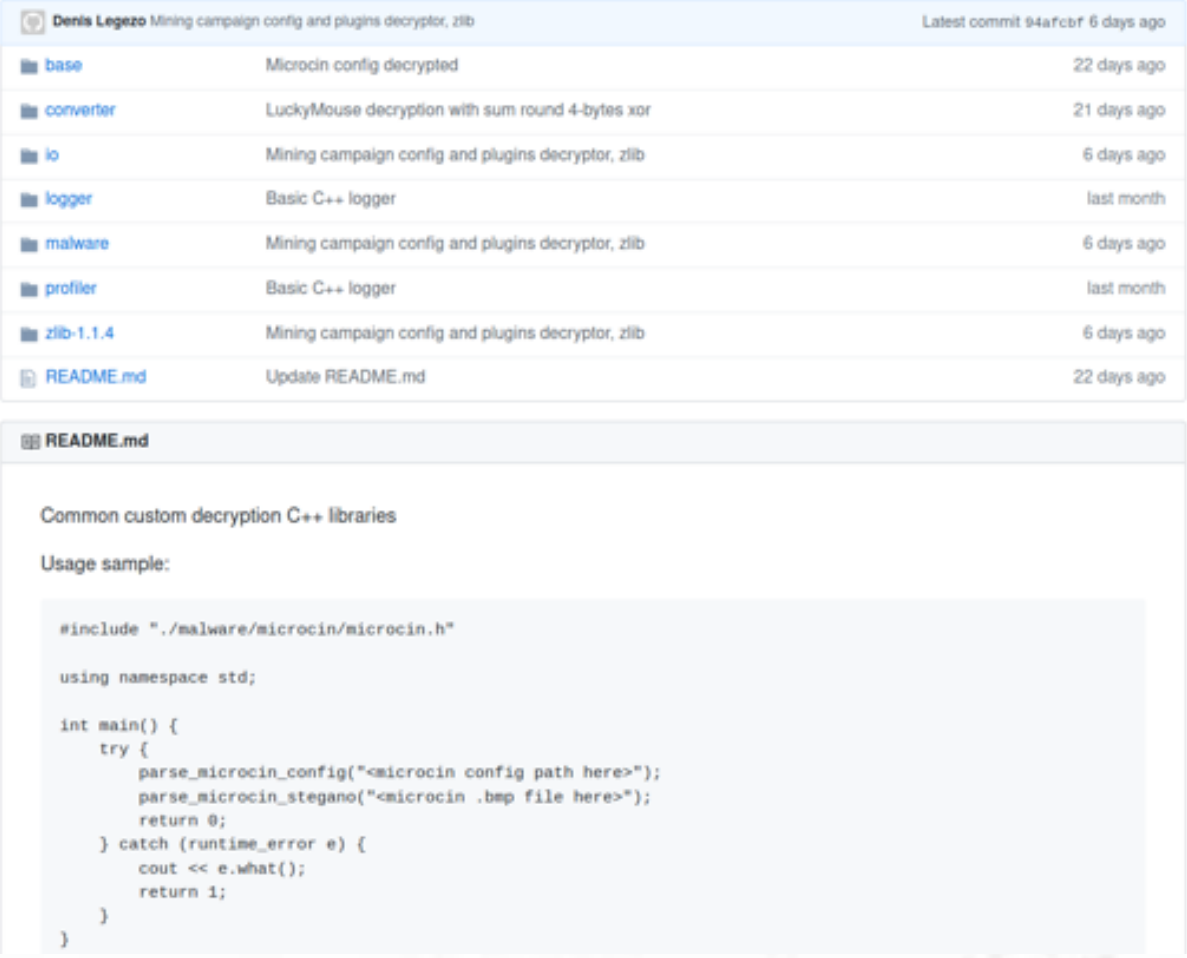
Second way pros

Knowledge separation

Real understanding

High re-usability

Pipe for dozens of samples



Denis Legezo Mining campaign config and plugins decryptor, zlib — Latest commit 94afcbf 6 days ago

base	Microcin config decrypted	22 days ago
converter	LuckyMouse decryption with sum round 4-bytes xor	21 days ago
io	Mining campaign config and plugins decryptor, zlib	6 days ago
logger	Basic C++ logger	last month
malware	Mining campaign config and plugins decryptor, zlib	6 days ago
profiler	Basic C++ logger	last month
zlib-1.1.4	Mining campaign config and plugins decryptor, zlib	6 days ago
README.md	Update README.md	22 days ago

README.md

Common custom decryption C++ libraries

Usage sample:

```
#include "./malware/microcin/microcin.h"

using namespace std;

int main() {
    try {
        parse_microcin_config("<microcin config path here>");
        parse_microcin_stegano("<microcin .bmp file here>");
        return 0;
    } catch (runtime_error e) {
        cout << e.what();
        return 1;
    }
}
```

First way pros

Speed for the first sample

May be you just don’t like to code

Far less error prone approach

Thank You!

HITBLOCKDOWN 002

livestream

Denis Legezo, @legezo, denis.legezo@kaspersky.com